

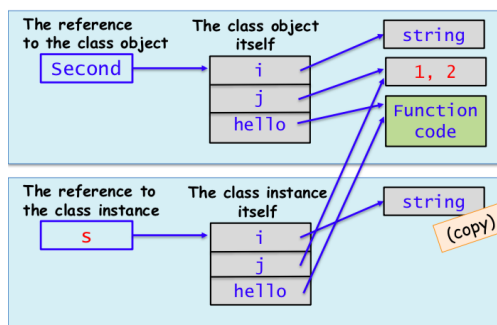
[Lecture 3] Classes, Inheritance and Virtual Methods

```
class Second:
    i = 'string'
    j = [1, 2]
    def hello():
        print('Hello')
```

- Creates a new **class object**
- Creates the new reference to that class object (in stack)
- Adds the reference to the current scope

Op:

1. Work with class attributes (`Second.j = [2, 3]`) and methods (`Second.hello()`)
2. Add new attributes (`Second.x = [1, 2]`)
3. Create class instances (`s = Second()`)



- New class instance **created**
- Contents of class object **gets copied** to the class instance
- For **immutable** data structures, **new copies are created**; for **mutable** data structures **no copies** are created

`s.hello()` $\Rightarrow$ `Second.hello(s)`

Incorrect, because hello() has no parameters

Constructor `__init__` -
method that called
automatically when
instance is created

Should always have at least one
parameter (ref to created
instance)

```
class Fourth:
    i = 12345
    def __init__(self):
        self.m1 = 777
        self.m2 = [1,2,3]
```

Adds two new members to the instance: m1
and m2

There is **no function
overloading** in Python

⇒

Classes can have **only one**
constructor

However

→

```
def __init__(self,k):
    if k>0:
        self.m1 = 777
        self.m2 = [1,2,3]
    else:
        self.x1 = { x, y }
        self.x2 = 'string'
```

Inheritance **single** and **multiple**

```
class B:
    mb = 333

class D(B):
    md = 777

b = B()
d = D()
```

Class D inherits
attributes and
methods (if any)
from class B + its
own

```
class B:
    def __init__(self,x):
        self.x = x

class D(B):
    def __init__(self,x,y):
        B.__init__(x)
        self.y = y

b = B(5)
d = D(5,7)
```

Конструктор наследуемого класса может вызывать
конструктор родительского

```
class B:
    mb = 333
    def f(self):
        return mb

class D:
    md = 777
    def f(self):
        return md
```

```
if someCondition:
    instance = B()
else:
    instance = D()
...
instance.f()
```

```
list = [ B(), D() ]
for i in list:
    i.f()
```

All class methods are by default **virtual**

All attributes & methods in classes are by default **public.**

(limited support for **private** attributes)

__N - by default **private**